REMARKS

This amendment is responsive to the Final Office Action dated October 5, 2005. Applicant has amended claims 1, 6, 7, 10, 18, 26, 35, 44, 60, 69, 78, 86, 94, 103, 112, 120 and 128. Applicant has added claim 137. Claims 1-137 are pending.

Amendments to Specification

Applicant has amended the specification to include an express statement of priority in accordance with the priority claim originally made when the pending application. The claim of priority was originally made in the filing papers accompanying the application, but was omitted from the specification. The Filing Receipt received from the USPTO indicates that the domestic priority claim was properly perfected and recognized by the USPTO. Applicant has attached a copy of the Filing Receipt as well as page 2 from the original filing transmittal showing the priority claim was properly made and recorded. Applicant submits that no fee for a late priority claim is due because the priority claim was properly made at the time of filing. Applicant respectfully requests that the Examiner enter the amendment.

Claim Rejection Under 35 U.S.C. § 103

In the Final Office Action, the Examiner rejected claims 1-6, 8-15, 17-20, 23 25-28, 31, 33, 34-37, 39-41, 43-49, 51-57, 59-65, 67, 68-74, 76, 77-83, 85-91, 101, 102-108, 110, 111-117, 119-125, 127-133., 135 and 136 under 35 U.S.C. 103(a) as being unpatentable over Nair et al. (US 2004/0193900) in view of Carpentier et al. (US 2005/0010792). Applicant respectfully traverses the rejection to the extent such rejections may be considered applicable to the claims as amended. The applied references fail to disclose or suggest the inventions defined by Applicant's claims, and provide no teaching that would have suggested the desirability of modification to arrive at the claimed invention.

Claims 1-17

In general, Applicant has amended claim 1 to clarify the differences between Applicant's described data transfer techniques and conventional peer-to-peer networks as well as conventional multicast delivery techniques, as described by the prior art cited by the Examiner.

Amended claim 1 is not taught or suggested by Nair in view of Carpentier for at least the following reasons:

First, the particular embodiment recited by amended claim 1 requires sending a request for data from a requesting computer to a targeted computer system and, in response, accessing a look-up list at the targeted computer system to identify other computers *that have previously requested and received* at least a portion of the requested data. In the Office Action, the Examiner asserts that Nair (0009-0010) teaches these elements. However, at 0009, Nair describes how in conventional peer-to-peer networks in which each computer (peer) broadcasts a respective list of music files to a centralized computer that maintains an index of potential <u>sources</u> for each file. Thus, Nair, like other conventional peer-to-peer networks, do not maintain a lookup-list of computers that *have previously requested and received data.* To the contrary, as expressly stated by Nair, each peer "broadcast its list" of files and the central computer maintains an index of files that lists the peers and the files at those peers. Typically, peer-to-peer networks do not track which peers have requested and received data, only those peers that are offering to <u>source</u> data. The peer computers in Nair broadcast available files, and that information is assembled into a centralized list. There is no teaching or suggestion of a list or other data structure that tracks which computers have requested and successfully received data. To the contrary, the index of Nair, like traditional peer-to-peer networks, only describes sources of files based on broadcast information and does not track past computers that have downloaded the files. For this reason, Nair in view of Carpentier provides no such teaching or suggestion of accessing at the targeted computer system a look-up list to identify other computers that have previously requested and received at least a portion of the requested data.

Second, amended claim 1 requires sending requests to the identified computers, wherein upon receiving the requests the identified computers have received *different partial portions* of the requested data. Claim 1 also requires encoding the different partial portions of the data at the identified computers in response to the requests, and sending the encoded different partial portions of the data from the identified computers to the requesting computer *prior to receiving remaining portions of the data.* Thus, the embodiment of claim 1, requires that at the time the requests are received by the set of identified computers (e.g., peers), the identified peer computers have received different partial portions of the data and that, prior to receiving the remaining portions, the identified peer computers encode and send the partial portions to the requesting computer.

With respect to previously presented claim 1, the Examiner cited Carpentier at 0069 as teaching other nodes simultaneously filling the missing parts of a file in a peer-to-peer network. In the cited passage, Carpentier describes a peer computer outputting a multicast request to a multicast group, and then receiving "chunks" of an asset from peers storing the asset. In particular, Carpentier at 0069 is referring to FIG. 5 that describes how a peer checks (receives) multicast transmissions from multiple sources, and stores portions of the multicast transmission that are needed.

However, unlike the embodiment recited in amended claim 1, the multiple-source multicast of Carpentier relies on multiples sources that have a desired file (asset). As described in Carpentier, in response to a request specifying a file (asset) name, multiple multicast sources output multicast transmissions of that file. Interested devices, i.e., those devices that have joined the multicast group, receive and store all or portions of the multicast transmissions as needed.

The embodiment of claim 1, however, requires that the identified computers (i.e., the sources) have *different partial portions of the requested data at the time the requests are received.* Moreover, amended claim 1 requires sending the encoded different partial portions of the data from the identified computers to the requesting computer *prior to receiving remaining portions of the data.* Unlike multi-source multicast described in Carpentier, claim 1 requires that the identified peer computers that source the requested data have different partial portions of the same data, e.g., a file, at the time a request is received, and that those source peer computers send encoded different partial portions prior to receiving remaining portions. This is not the case with conventional multi-source multicast where each source sourcing a multicast transmission of data has a complete copy of that data.

Thus, although Carpentier describes use of multicast to "fill in missing parts" of a file, as suggested by the Examiner, Carpentier fails to teach or suggest a transfer mechanism in which peer computing devices sourcing requested data only have different partial portions of that data, as required by the embodiment of Applicant's amended claim 1. For this reason, Nair in view of Carpentier provides no teaching or suggestion of a process in which peer computers send encoded different partial portions of the data to a requesting computer prior to receiving remaining portions of the data, possibly from other peer devices. To the contrary, the peer devices acting as multi-source multicast devices in Carpentier respond to broadcast requests listing specific asset names and, apparently, output multicast transmissions for those assets. There is no teaching or suggestion that the peers in Carpentier operate in any manner other than conventional multicast, where sources of a multicast group maintain copies of files and output multicast transmissions for those files in response to requests (joins).

With respect to claim 4, the Examiner correctly recognizes that Nair in view of Carpentier fails to teach or suggest a look-up list that is populated with nodes based on data transfer rates. However, the Examiner takes Official Notice that such elements are known within peer-to-peer networks. Applicant submits that, as described above, conventional peer-to-peer networks do not track peer devices that have requested and successfully downloaded files. To the contrary, as described by Nair, peer-to-peer systems maintain a central index of devices acting as sources for files. Those devices may not have event downloaded a particular file, and many peers that did download a file may not necessarily act as a source for those files. Thus, Applicant submits that conventional peer-to-peer networks fail to maintain a look-up list of computers that have previously requested and received at least a portion of the requested data, where the list is populated with nodes based on transfer rates, as required by claim 4. In accordance with MPEP 2144.03 (C), Applicant respectfully requests the Examiner provide documentary evidence if the rejection is to be maintained.[1]

With respect to claim 6, Applicant has amended the claim to require that the look-up list is a mesh list that record which partial portions of the data each of the other computers has received. Claim 6 as amended further requires that accessing the look-up list to identify the other computers includes selecting the identified other computers based on the record of which partial portions of the data each of the other computers has received. With respect to previously presented claim 6, the Examiner asserts that Nair at 0038 teaches a mesh list. However, this portion of Nair describes a console player unit having media files and does not describe a mesh

[1] See MPEP 2144.03 (C) (stating that if applicant adequately traverses the examiner's assertion of official notice, the examiner must provide documentary evidence in the next Office action if the rejection is to be maintained. See 37 CFR 1.104(c)(2). See also Zurko, 258 F.3d at 1386, 59 USPQ2d at 1697 ("[T]he Board [or examiner] must point to some concrete evidence in the record in support of these findings" to satisfy the substantial evidence test). If the examiner is relying on personal knowledge to support the finding of what is known in the art, the examiner must

list. For at least these reasons, Nair in view of Carpentier does not teach or suggest a mesh list that records which partial portions of the data each of the other computers has received. Moreover, Nair in view of Carpentier does not teach accessing the look-up list to identify the other computers includes selecting the identified other computers based on the record of which partial portions of the data each of the other computers has received.

With respect to claim 7, the Examiner acknowledges that Nair in view of Carpentier does not teach or suggest encoded using an acknowledgement independent equalized data packet encoding scheme that is a FEC encoding. However, the Examiner states that Schuster teaches using a FEC scheme for encoding data, and that it would be obvious to one of skill in the art to modify Nair-Carpentier in view of Shuster.

Schuster makes reference to the use of forward error correction (FEC) when transferring voice data from a source node (e.g., a network telephony device) to a destination node (e.g., a second network telephony device) via one or more packet streams. Thus, Schuster describes the use of forward error correction for individual TCP/IP sessions between telephony devices. Applicant submits that modification of Nair-Carpentier in view of Schuster, as suggested by the Examiner, would result in a peer-to-peer network that uses a multicast delivery mechanism having forward error correction. Nair-Carpentier in view of Schuster fails to teach or suggest a network transfer where the multiple, different identified computers having only partial portions of requested data re-encode those partial portions for transmission to the requesting computer using encoding using an acknowledgement independent equalized data packet encoding scheme that is a FEC encoding, as required by amended claim 7.

For these or other similar reasons, the cited references fail to establish a prima facie case for non-patentability of Applicant's claims 1-17 under 35 U.S.C. § 103(a) as amended. Withdrawal of these rejections is requested. Applicant points out that the MPEP 2144 makes clear that while "official notice" may be relied on, these circumstances should be <u>rare</u> when an application is under final rejection or action under 37 CFR 1.113.

Claims 18-25

provide an affidavit or declaration setting forth specific factual statements and explanation to support the finding. See 37 CFR 1.104(d)(2).)

Applicant has amended independent claim 18 to require that a source computer maintain a list of peer computers that have previously downloaded at least a portion of the data, and that the list record which partial portions of the data each of the other computers has received. Claim 18 as amended further requires that accessing the look-up list to identify the other computers includes selecting the identified other computers based on the record of which partial portions of the data each of the other computers has received.

As discussed above, Nair describes how in conventional peer-to-peer networks in which each computer (peer) broadcasts a respective list of music files to a centralized computer that maintains an index of files that lists the potential sources for each file. Thus, Nair, like other conventional peer-to-peer networks, do not maintain at lookup-list of computers that *have previously requested and received data.*

Further, Nair in view of Carpentier does not teach or suggest a list that records which partial portions of the data each of the other peer computers has received. Moreover, Nair in view of Carpentier does not teach or suggest selecting the identified other computers based on the record of which partial portions of the data each of the other computers has received.

Moreover, Nair in view of Carpentier and Shuster does not teach or suggest encoding the different portions of the data at the peer computers using an acknowledgement independent equalized data packet encoding scheme prior to receiving all of the data at the peer computers. With respect to these elements, the Examiner refers to Nair at 0045. At 0045 Nair describes a central media web server 430 having a real-time encoding and compression system 440. Nair fails to describe using an acknowledgement independent equalized data packet, and Nair in view of Carpentier and Shuster fail to suggest a network transfer mechanism where different portions of the data are re-encoded at the peer computers using an acknowledgement independent equalized data packet encoding scheme prior to receiving all of the data at the peer computers

Schuster makes reference to the use of forward error correction (FEC) when transferring voice data from a source node (e.g., a network telephony device) to a destination node (e.g., a second network telephony device) via one or more packet streams. Thus, Schuster describes the use of forward error correction for individual TCP/IP sessions between telephony devices. Applicant submits that modification of Nair-Carpentier in view of Schuster, as suggested by the Examiner, would result in a peer-to-peer network that uses a multicast delivery mechanism having forward error correction. Nair-Carpentier in view of Schuster fails to teach or suggest a network transfer where the multiple, different peer computers having only partial portions of requested data re-encode those partial portions for transmission using encoding using an acknowledgement independent equalized data packet encoding scheme, as required by amended claim 18.

Claims 26-136

For these or other similar reasons, the cited references fail to establish a prima facie case for non-patentability of Applicant's claims 26-136 under 35 U.S.C. § 103(a) as amended. Withdrawal of these rejections is requested.

Objective Evidence and Secondary Considerations

To aid the Examiner in appreciating the novelty and non-obviousness of Applicant's claimed inventions, Applicant submits a recently published paper from researchers at Georgia Institute of Technology and Microsoft Research entitled Network Coding for Large Scale Content Distribution.[2] This paper provides evidence that demonstrates that industry has only recently realized the advantages of using independent network coding. Moreover, the paper compares the approach with conventional tree-based delivery schemes, such as multicast. Applicant requests the Examiner consider this objective evidence as further establishing the novel and non-obviousness of Applicant's claimed invention.[3]

New Claims:

Applicant has added new independent claim 137 to the pending application. The applied references fail to disclose or suggest the inventions defined by Applicant's new claims, and provide no teaching that would have suggested the desirability of modification to arrive at the claimed inventions.

For example, claim 137 requires receiving at a source computer requests for data from a first requesting computer and a second requesting computer and, in response to the requests, delivering a first encoded portion of data from the source computer to the first requesting

[2] IEEE Infocom, 2005 (http://research.microsoft.com/~pablo/papers/nc_contentdist.pdf).

[3] MPEP 716.01.

computer and a second encoded portion of the data from the source computer to the second requesting computer, wherein the first encoded portion and the second encoded portions are encoded using a forward error correction (FEC) encoding. Claim 137 further requires exchanging the first encoded portion and the second encoded portion of the data between the first requesting computer and the second requesting computer; and decoding the first encoded portion and the second encoded portion of the data to recreate the requested data at the first requesting computer and the second requesting computer.

Unlike claim 137, the cited references fail to describe a network data transfer mechanism where requesting computers having only partial portions of the requested data exchange those portions to recreate the requested data. As described above, there is no teaching or suggestion that the peers in Carpentier operate in any manner other than in conformance with conventional multicast, where sources of a multicast group maintain copies of files and output multicast transmissions for those files in response to requests (joins).

No new matter has been added by the new claims. Support can be found, for example, throughout the description with respect to FIG. 2 and, in particular, nodes 2202d and 2202e.

CONCLUSION

All claims in this application are in condition for allowance. Applicant respectfully requests reconsideration and prompt allowance of all pending claims. Please charge any additional fees or credit any overpayment to deposit account number 50-1778. The Examiner is invited to telephone the below-signed attorney to discuss this application.

Date:
February 6, 2006
SHUMAKER & SIEFFERT, P.A.
8425 Seasons Parkway, Suite 105
St. Paul, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

By:

Name: Kent J. Sieffert
Reg. No.: 41,312

OIPE IAP59
FEB 10 2006
PATENT & TRADEMARK OFFICE

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
WASHINGTON, D.C. 20231
www.uspto.gov

APPLICATION NUMBER	FILING DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	DRAWINGS	TOT CLAIMS	IND CLAIMS
10/033,305	12/28/2001	2121	1960	4110-4002US1	3	136	16

CONFIRMATION NO. 8199

FILING RECEIPT

OC000000007399848

MORGAN & FINNEGAN, L.L.P.
345 Park Avenue
New York, NY 10154-0053

Date Mailed: 01/31/2002

Receipt is acknowledged of this nonprovisional Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please write to the Office of Initial Patent Examination's Customer Service Center. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).**

MORGAN & FINNEGAN LLP
2002 FEB 11 P 1:03

Applicant(s)

Justin Falconer Chapweske, Lauderdale, MN;

Domestic Priority data as claimed by applicant

THIS APPLN CLAIMS BENEFIT OF 60/258,981 12/29/2000

Foreign Applications

If Required, Foreign Filing License Granted 01/30/2002

Projected Publication Date: Request for Non-Publication Acknowledged

Non-Publication Request: Yes

Early Publication Request: No

**** SMALL ENTITY ****

CASE 4110-4002US I ATTY JK
INFORMATION DISCLOSURE STATEMENT March 28, 2002 (3M
FOREIGN FILING ~~Sept 28, 2002~~
CONVENTION DATE EXPIRES ~~Dec. 28, 200~~

Title

Apparatus, method, and system for an acknowledgement- independent equalized data packet transfer mechanism over a peer to peer network

Preliminary Class

700

LICENSE FOR FOREIGN FILING UNDER
Title 35, United States Code, Section 184
Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Office of Export Administration, Department of Commerce (15 CFR 370.10 (j)); the Office of Foreign Assets Control, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

OIPE IAP59
FEB 1 0 2006
PATENT & TRADEMARK OFFICE

☒ **REQUEST AND CERTIFICATION UNDER 35 U.S.C. §122(b)(2)(B)(i) (form PTO/SB/35)**

As indicated on the attached Request and Certification. Applicant(s) certify that the invention disclosed in the attached application HAS NOT and WILL NOT be the subject of an application filed in another country, or under a multilateral agreement, that requires publication at eighteen months after filing. Applicant(s) therefore request(s) that the attached application NOT be published under 35 U.S.C. §122(b).

□ Incorporation by Reference:

□ The entire disclosure of the prior application, from which a copy of the combined Declaration and Power of Attorney is supplied herein, is considered as being part of the disclosure of the accompanying application and is incorporated herein by reference.

□ Deletion of Inventors (37 C.F.R. §1.63(d) and §1.33(b)

Signed statement attached deleting inventor(s) named in the prior application serial no. ______, filed ______.

□ Microfiche Computer Program (Appendix)

□ page(s) of Sequence Listing

□ computer readable disk containing Sequence Listing

□ Statement under 37 C.F.R. §1.821(f) that computer and paper copies of the Sequence Listing are the same

☒ Assignment Papers (assignment cover sheet and assignment documents)

□ A check in the amount of $40.00 for recording the Assignment

☒ Charge the Assignment Recordation Fee to Deposit Account No. 13-4500, Order No. 4110-4002US1.

□ Assignment Papers filed in the parent application Serial No. ______

□ Certification of chain of title pursuant to 37 C.F.R. §3.73(b)

□ Priority is claimed under 35 U.S.C. §119 for:
Application No(s). ______ , filed______,in ______(country).

□ Certified Copy of Priority Document(s) [______]

□ filed herewith

□ filed in application Serial No. ______, filed ______.

□ English translation document(s) [______]

□ filed herewith

□ filed in application Serial No. ______,filed ______.

☒ Priority is claimed under 35 U.S.C. §119(e) for:
Provisional Application No. 4110-4002US1, filed 12/29/00.

Network Coding for Large Scale Content Distribution

IEEE Infocom 2005

Christos Gkantsidis
College of Computing
Georgia Institute of Technology
Atlanta, GA, 30332, USA
Email: gantsich@cc.gatech.edu

Pablo Rodriguez Rodriguez
Microsoft Research
Cambridge, CB3 0FD, UK
Email: pablo@microsoft.com

Abstract— **We propose a new scheme for content distribution of large files that is based on network coding. With network coding, each node of the distribution network is able to generate and transmit encoded blocks of information. The randomization introduced by the coding process eases the scheduling of block propagation, and, thus, makes the distribution more efficient. This is particularly important in large unstructured overlay networks, where the nodes need to make block forwarding decisions based on local information only. We compare network coding to other schemes that transmit unencoded information (i.e. blocks of the original file) and, also, to schemes in which only the source is allowed to generate and transmit encoded packets.**

We study the performance of network coding in heterogeneous networks with dynamic node arrival and departure patterns, clustered topologies, and when incentive mechanisms to discourage free-riding are in place. We demonstrate through simulations of scenarios of practical interest that the expected file download time improves by more than 20-30% with network coding compared to coding at the server only and, by more than 2-3 times compared to sending unencoded information. Moreover, we show that network coding improves the robustness of the system and is able to smoothly handle extreme situations where the server and nodes leave the system.

I. Introduction

Typical content distribution solutions are based on placing dedicated equipment inside or at the edge of the Internet. The best example of such solutions is Akamai [1], which runs several tens of thousands of servers all over the world. In recent years, a new paradigm for Content Distribution has emerged based on a fully distributed architecture where commodity PCs are used to form a cooperative network and share their resources (storage, CPU, bandwidth).

Cooperative content distribution solutions are inherently self scalable, in that the bandwidth capacity of the system increases as more nodes arrive: each new node requests service from, and, at the same time, provides service to other nodes. Because each new node contributes resources, the capacity of the system grows as the demand increases, resulting in limitless system scalability. With cooperation, the source of the file, i.e. the server, does not need to increase its resources to accommodate the larger user population; this, also, provides resilience to "flash crowds"— a huge and sudden surge of traffic that usually leads to the collapse of the affected server. Therefore, end-system cooperative solutions can be used to efficiently and quickly deliver software updates, critical patches, videos, and other large files to a very large number of users while keeping the cost at the original server low.

The best example of an end-system cooperative architecture is the BitTorrent system, which became extremely popular as a way of delivering the Linux distributions and other popular content. BitTorrent splits the file into small blocks, and immediately after a node downloads a block from the origin server or from another peer, the node behaves as a server for that particular block, and, thus, contributes resources for serving the block. Further design choices, such as intelligent selection of the block to download, and parallel downloading of blocks, improve the performance of the system. For a detailed description of the BitTorrent system see [2].

Despite their enormous potential and popularity, existing end-system cooperative schemes such as BitTorrent, may suffer from a number of inefficiencies which decrease their overall performance. Such inefficiencies are more pronounced in large and heterogeneous populations, during flash crowds, in environments with high churn, or when cooperative incentive mechanisms are in place. In this paper we propose a new end-system cooperative solution that uses *network coding*, i.e. data encoding at the interior nodes of the network, to overcome most of these problems.

A. Network Coding

Network coding is a novel mechanism proposed to improve the throughput utilization of a given network topology [3]. The principle behind network coding is to allow intermediate nodes to encode packets. Compared to other traditional approaches (e.g. building multicast trees), network coding makes optimal use of the available network resources and, moreover, computing a scheduling scheme that achieves such rate is computationally easy. An overview of network coding and a discussion of possible Internet applications is given in [4].

Using ideas borrowed from network coding, we propose an end-system content distribution solution which optimally uses the resources of the network. Every time a client needs

This work was done while the first author was with Microsoft research.

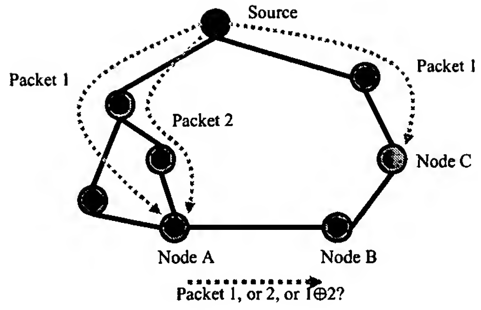

Fig. 1. Network Coding benefits when nodes only have local information.

to send a packet to another client, the source client generates and sends a linear combination of all the information available to it (similarly to XORing multiple packets). After a client receives enough linearly independent combinations of packets, the client can reconstruct the original information.

In a large distributed cooperative system finding an optimal packet propagation scheme that minimizes the client download time is very difficult. This is especially the case in practical systems that cannot rely on a central scheduler and, instead, allow nodes to make local decisions. The scheduling problem becomes increasingly difficult as the number of nodes increases, when nodes are at different stages in their downloads, or when incentive mechanisms are introduced to prevent leeching clients. As we will see in this paper, network coding makes efficient propagation of information in a large scale distributed system with no central scheduler easier, even in the scenarios described above.

To illustrate how network coding improves the propagation of information without a global coordinated scheduler we consider the following (simple) example. In Figure 1 assume that Node A has received from the source packets 1 and 2. If network coding is not used, then, Node B can download either packet 1 or packet 2 from A with the same probability. At the same time that Node B downloads a packet from A, Node C independently downloads packet 1. If Node B decides to retrieve packet 1 from A, then both Nodes B and C will have the same packet 1 and, the link between them can not be used.

If network coding is used, Node B will download a linear combination of packets 1 and 2 from A, which in turn can be used with Node C. Obviously, Node B could have downloaded packet 2 from A and then use efficiently the link with C. However, without any knowledge of the transfers in the rest of the network (which is difficult to achieve in a large, complex, and distributed environment), Node B cannot determine which is the right packet to download. On the other hand, such a task becomes trivial using network coding. It is important to note that the decision on which packets to generate and send does not require extra information about the downloads in the rest of the network; thus the content distribution effort is greatly simplified.

B. Contributions

We now summarize the main contributions of this paper:

1) We describe a practical system based on network coding for file distribution to a large number of cooperative users. Our approach does not require knowledge of the underlying network topology, and, in addition, nodes make decisions of how to propagate packets based on local information only. Notice that typical network coding schemes assume both knowledge of the underlying topology and centralized decisions about how to encode and propagate content. Compared to other content distribution schemes, our network coding based approach eases the problem of block propagation scheduling across a large scale distributed system and makes better use of the available network capacity.

2) We provide experimental evidence that, in many situations of practical interest, network coding performs better than transmitting unencoded blocks, or using techniques that are based on erasure codes, which can be thought as coding but only at the server. Network coding performs better by almost a factor of two compared to performing encoding at the server and by a factor of three compared to not coding at all when the topology is clustered. Similarly, network coding improves the download rates by almost 20% compared to source coding and by more than 30% compared to no coding in an heterogeneous network. During the early stages of a flash crowd, network coding outperforms source coding and no coding by 40% and 200% respectively. Even when the system reaches a steady-state, network coding still provides significant benefits compared to using other techniques. Moreover, when tit-for-tat incentives are used the performance of network coding is barely impacted, while, other schemes suffer significantly.

3) We show that our network coding system is very robust to extreme situations with sudden server and node departures. By using network coding, nodes are able to make progress and finish a download even if the server leaves shortly after uploading only one copy of the file to the system and nodes depart immediately after they finish their download. Without network coding, if both the server and some peers suddenly depart the system, some blocks of the original file or of the source-encoded file will disappear and the remaining nodes will not be able to finish their downloads. This demonstrates that with network coding nodes are able to finish their downloads even in extreme circumstances.

The rest of the paper is organized as follows. In Section II we provide an overview of related work. In Section III we describe our model for end-system cooperative content distribution and discuss how network coding can be used as a building block of such a system. In Section IV we provide experimental evidence of the benefits of using network coding over sending blocks of the original file and over source coding. We summarize in Section V and discuss some open problems.

II. Related Work

Implementing large scale content distribution using end-system cooperation techniques has been the focus of multiple research efforts during the recent years.

a) Tree-Based Cooperative Systems: Two of the first proposals to use end-systems for content distribution were Yoid and EndSystem multicast, which provided an end-system overlay multicast architecture for the delivery of video and audio content [5], [6]. Following a similar concept, SplitStream [7] builds multiple overlay multicast trees to enable a more robust and fair system where all nodes contribute roughly the same to the distribution effort and the departure of a node causes minor disruptions. Similarly, Coopnet [8] forms either random or deterministic node-disjoint trees, and it includes a Multiple Description Coding layer (MDC) [9] to maximize the streaming quality experienced by the users.

Creating and maintaining shortest-path multicast trees provides an optimized architecture for the delivery of real-time streaming content. However, architectures that employ tree topologies are bandwidth-limited in that the transfer rate to a client will only be as fast as the throughput of the bottleneck link on the path from the server, and moreover, "perpendicular" connections among nodes are often hard to exploit. For file downloads, optimizing bandwidth is often more critical than optimizing delay, and therefore, tree-based architectures may not always be the best approach.

b) Mesh Cooperative Architectures: As an alternative to tree-based systems, a number of mesh architectures have also been suggested. Mesh cooperative architectures can substantially improve the download rates by using additional connections, and, hence, additional bandwidth and better opportunities for finding content of interest, and *parallel-downloads* [10], [11].

However, one major drawback of mesh cooperative architectures is that since no trees are constructed, there are no pre-determined distribution paths for the content. Instead, the topology includes multiple parallel paths over which content should propagate. If nodes make local decisions, then it is possible that the same block travels over multiple competing paths and, hence, network resources are under-utilized and the download rates decrease.

The most popular of such cooperative architectures is BitTorrent [2]. A detailed analysis of BitTorrent's performance can be found in [12], [13]. BitTorrent provides an end-system cooperative architecture to facilitate fast downloads of popular files.

To improve the efficient propagation of content among nodes, BitTorrent uses a rarest-first block download policy. Such policy attempts a uniform distribution of blocks in the network and, as a result, nodes have better chances of finding missing blocks and finishing the download faster. Still, however, nodes that are close to finishing their download, may attempt to obtain the missing blocks from the server, and, as a result, cause unnecessary server overloading. To overcome this problem, Slurpie [14] proposes a randomized back off strategy combined with an effective group size estimator that precisely controls the load on the server.

c) Erasure Codes: A number of cooperative architectures [15], [16] have proposed the use of Erasure Codes[1] [17], [18] (e.g. Digital Fountain) to efficiently transfer bulk data. The digital fountain approach enables end-hosts to efficiently reconstruct the original content of size n from roughly a subset of any n symbols from a large universe of encoded symbols. However, since the sets of symbols acquired by nodes are likely to overlap substantially, care must be taken to enable them to collaborate effectively. This makes cooperation and reconciliation among nodes more difficult than when content is unencoded. To overcome this problem [16] proposes techniques to efficiently reconciliate encoded content among different nodes using sketches, bloom filters, etc.

While encoded content can efficiently handle losses, asynchronous arrivals, and churn, locating missing data items may still be a challenge. To address the issue of efficiently locating useful encoded blocks within the system, Bullet [15] proposes a mechanism that periodically disseminates summaries of data sets uniformly sampled over a random subset of global participants.

d) Network Coding: Network coding was first considered in the pioneering work by Alswede et al. [3], where they showed that a sender can communicate information to a set of receivers at the broadcast capacity of the network provided one allows network coding.

High utilization of a given topology can also be achieved using multiple edge-disjoint distribution trees (especially in the case where all nodes are receivers). In fact, several schemes (e.g. SplitStream, CoopNet) have proposed to utilize multiple multicast trees (forest) to deliver striped data from the source to all receivers. These proposals can indeed improve end-to-end throughput beyond that of a single tree, however, computing the strategies to achieve optimal throughput using multiple trees has been shown to be NP-complete and APX-hard [19]–[21] Instead, recent studies have shown that network coding can significantly facilitate the design of efficient algorithms to compute and achieve such optimal throughput [22].

Most of the previous work on network coding is largely based on theoretical calculations that assume a detailed knowledge of the topology, and a centralized knowledge point for computing the distribution scheme. However, little effort has been made to study the practical aspects of implementing network coding on a real distributed setting. In this regard [22] considers the feasibility of applying the theoretical insights in network coding to increasing throughput in actual multicast systems over wide-area networks. In [23], Chou et al. propose a practical network coding system for streaming content. Similarly, in [24], K. Jain et al. provide (contemporaneously with this work) analytical evidence that supports the use of network coding in peer-to-peer networks. Based on the work presented in [23] and [24], we propose a practical end-system cooperative architecture that uses network coding to enable efficient large scale content distribution.

[1] we use the terms Erasure Codes, FEC, and Source Coding interchangeably across the paper

Network coding can be seen as an extension or generalization of the Digital Fountain approach since both the server and the end-system nodes perform information encoding. Note, however, that restricting erasure codes only to the origin server implies that intermediate nodes can only copy and forward packets. This results in *the same* erasure codes being blindly copied over from one node to another without knowing whether they will be of use downstream. Again, the existence of cycles in the topology will result in multiple copies of the same block arriving at a given receiver through different paths, and as a result the effective capacity of the system decreases. With Network Coding, on the other hand, a given code is combined with many other codes as it propagates through the network, thus, drastically reducing the probability of a code not being useful to a given node.

When the total number of different erasure codes in the system is very high, then, the probability of finding *informative* codes increases and server encoding performs similarly to network coding. However, when the server capacity is limited, when nodes leave the system soon after they finish the download, or when some coded blocks do not efficiently propagate through the network, the total number of *different* codes in the system is significantly reduced and using erasure codes has far less clear benefits compared to not performing encoding at all.

III. Model

In this section, we describe our model for end-system cooperative content distribution. This model can be used to either distribute blocks of the original file (no coding), or blocks of encoded information where the encoding can happen either only at the source (source coding), or both at the source and at the network (network coding). We will outline the basic operation of this system, emphasizing some algorithmic parameters that affect its performance. However, a detailed protocol implementation is outside of the scope of this paper. These algorithmic parameters and their impact will be studied in Section IV.

A. Model of a collaborative content distribution network

We assume a population of users[2] that are interested in retrieving a file, which originally exists in a single server. The capacity of the server is limited (a server could be an end-user) and, thus, users contribute their bandwidth resources to help other users. Since the server does not have the capacity to serve all users simultaneously, it divides the file into k blocks and uploads blocks at random to different clients. The clients collaborate with each other to assemble all the k blocks to reconstruct the original file. This is very similar to how current end-cooperative systems, especially BitTorrent, work.

We assume that users do not know the identities of all other users; they only know the identities of a small subset of them, which we call the neighborhood. We assume that the neighboring relation is symmetric, i.e. if node A is in the neighborhood of B, then, also, B is in the neighborhood of A. Each node can exchange information, which includes blocks of the file and other protocol messages, only with its neighbors. The size of the neighborhood is normally a small value (e.g. 4-6).

The way nodes join the network is as follows. Upon arrival, each user will contact a centralized server that will provide a random subset of other users already in the system (similar to the tracker concept in [2]). The new user will then connect to each of them to construct its neighborhood. Content flows along edges in the resulting overlay topology. Rather than using a centralized server, other mechanisms for providing random subsets of nodes can be used like the ones proposed in [25], [26].

In the case that some nodes lose some of their neighbors (because they left the system), or when a node needs to use more neighbors to improve its parallel download rate, the node can request additional neighbors.

In this work, we assume that the major bottleneck in the system is the capacity of the access link of each user (and of the server). The total rate by which a user can receive information from all its neighbors is limited by the download capacity of the user; similarly, the total rate by which the user can upload information to all its neighbors is limited by the upload capacity of the user. For the purpose of this paper we assume symmetric links, where the download capacity is equal to the upload capacity of a node and both capacities are independent. We have experimented with asymmetric access capacities and observed very similar results and thus we omit the details of this case.

B. Content propagation of uncoded and source-encoded information

Each time there is a transfer of a block from the server or a user to another user, a decision process is involved as to which block will be downloaded. We assume that neither the server, nor any user have a complete information about the blocks that each user in the system has. Instead, each user only knows about the blocks it has downloaded and the blocks that exist in its neighbors and, thus, the algorithm for deciding which block to transfer is based on *local information* only. In our system, we have experimented with the following heuristics, which are commonly used in current systems:

- *Random block.* The block to be transfered is decided at random among the blocks that exist in the source (if the source is the server, then a random block among all blocks).
- *Local Rarest.* The block to be transfered is picked among the rarest block in the neighborhood. If there are multiple rarest blocks, a block at random is picked among them.
- *Global Rarest.* This is a baseline scheme which is not practical in large networks. The block to be transfered is the system-wise rarest among all blocks that exists in the neighborhood. This is a heuristic that gives priority to very rare blocks in the hope of improving the performance of the system.

[2] we use the terms nodes and users interchangeably across the paper

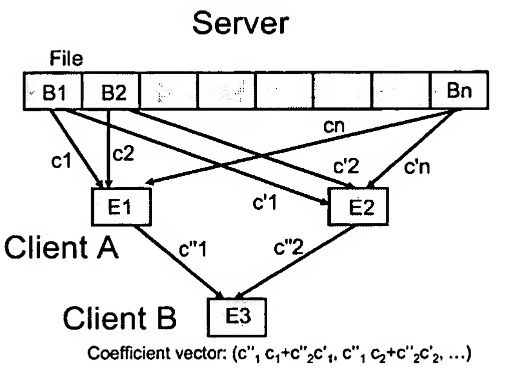
Server
File
B1
B2
Bn
c1
c2
cn
c'2
c'n
c'1
E1
E2
Client A
c"1
c"2
Client B
E3
Coefficient vector: (c"$_1$ c$_1$+c"$_2$c'$_1$, c"$_1$ c$_2$+c"$_2$c'$_2$, ...)

Fig. 2. Sample description of our network coding system.

The BitTorrent system uses a combination of the *Random* and *Local Rarest* schemes. In the beginning each nodes uses *Random* and after a few blocks have been downloaded it switches to *Local Rarest.*

When server coding is used, the system works very similar to the description above. However, the server gives blocks of encoded information and not of the original file. The server generates a new encoded block each time it needs to upload a block to a user. The user needs to download k distinct blocks, either from the server or from other peers, to reconstruct the original file, where k is the number of the blocks of the original file. This is an ideal example of a Forward Error Correction (FEC) encoding scheme. Typical FEC schemes, such as Reed-Solomon codes, generate a constant number of extra encoded packets. Rateless codes generate a practically infinite number of encoded packets, but they require slightly more than k packets for reconstruction.

C. Content propagation with network coding.

In the case of network coding, both the server and the users perform encoding operations. Whenever a node or the server needs to forward a block to another node, it produces a linear combination of all the blocks it currently stores. The operation of the system is best described in the example of Figure 2.

Assume that initially all users are empty and that user A contacts the server to get a block. The server will combine all the blocks of the file to create an encoded block $E1$ as follows. First, it will pick some random coefficients $c_1, c_2, \ldots, c_n$, then multiply each element of block i with c_i, and finally add the results of the multiplications together. All these operations take place in a finite field. Observe that the probability that two independent nodes with the same set of blocks pick the same set of coefficients and construct the same block depends on the size of the field. If the field is very small such "collisions" may happen and they will reduce the performance of the system. In most practical cases a field size of 2^{16} should be enough. (For a related discussion of the size of the field and the probability of decoding in a randomized setting see [27].)

The server will then transmit to user A the result of the addition, $E1$ and the *coefficient vector* $\vec{c} = (c_i)$. Assume now that user A has also another block of encoded information $E2$, either directly from the server or from another peer, with its associated vector of coefficients. If user A needs to transmit an encoded block $E3$ to user B, A generates a linear combination of its two blocks $E1$ and $E2$ as follows. User A picks two random coefficients c''_1 and c''_2, multiplies each element of block E_1 with the coefficient c''_1 and similarly for the second block E_2, and adds the results of the multiplication. The block transmitted to user B, say $E3$, is equal to $c''_1 \cdot E_1$ and $c''_2 \cdot E_2$. The coefficient vector $\vec{c''}$ associated with the new block is equal to $c''_1 \cdot \vec{c} + c''_2 \cdot \vec{c'}$.

Observe that a node can recover the original file after receiving k blocks for which the associated coefficient vectors are *linearly independent* to each other. The reconstruction process is similar to solving a system of linear equations.

The benefit we expect to get by using network coding is due to the randomization introduced each time we generate a new encoded block. Recall that without network coding, each user needs to decide which block to receive based on a local decision. This may be a suboptimal decision since such block may be already well represented in other areas of the network. On the other hand, with network coding, we perform a linear combination of all available blocks at a given node. Both popular as well as unpopular blocks are combined into a single block without having to estimate their popularity. If at least one of the combined blocks is of use to other nodes down the path, then the linear combination will also be useful. The only way that a generated encoded block is not useful is if the same block was generated independently elsewhere in the system (or from the same node in a previous transmission). However, as blocks traverse the network they get combined with other blocks in different nodes, making the probability of that happening particularly small.

Observe that in the case of network coding we do not have to worry about how to pick the block to transmit to the other node; we combine all the available blocks. However, the receiver still needs to decide whether the blocks at the server encode information not available at the receiver. Recall that it is possible to generate many linear encodings of the same information (if the coefficient vectors span the same space). If there is at least one packet at the sender that is not spanned by the coefficient vectors available at the receiver, then a random linear encoding at the sender will (almost always) produce a block that is of interest to the receiver; we call such blocks *innovative.* A simple approach to check for the possibility of generating innovative blocks is to ensure that each node knows the coefficient vectors of its neighbors. By using the neighbors' coefficients and its own coefficients, a given node can easily calculate the rank of the combined matrices and determine which nodes can provide new blocks and moreover how many blocks they can provide.

An alternative and cheaper approach is to have the sender generate a random linear combination of the available (to the sender) coefficient vectors and send the resulting coefficient vector to the receiver. If the receiver determines that the received vector is a linear combination of the vectors already available at the receiver, then it assumes that the sender does not have innovative blocks to send and waits for future updates from the sender. Observe that an unlucky choice of the random coefficients may lead the receiver to conclude that the server does not have innovative information, when in reality it does; such unlucky events should be very rare.

Note that the overhead of transmitting the coefficient vectors is quite small. In most practical scenarios, the size of each block is normally in the order of several hundreds of KBytes [10] whereas the size of a coefficient vector is smaller than one packet.

D. Incentive Mechanisms

An important problem of current collaborative content distribution networks is free-riding; many users take advantage of the resources offered to the network by other users without contributing their own resources. Free-riding can seriously degrade the performance of the content distribution [28], and, as a result, many collaborative networks have introduced mechanisms to discourage free-riding.

In our system we use two mechanisms to discourage free riding. The first is that we give priority to exchanges over free uploading to other nodes. In other words, when there is contention for the upload capacity of a user, the user will preferentially upload blocks of information to neighbors from which it is also downloading blocks. Thus, the nodes allocate their capacity preferentially to mutual exchanges and then use the remaining upload capacity for free downloads.

The second incentive mechanism that we use is inspired by the tit-for-tat approach used in the BitTorrent network [2]. A user does not upload content to another user unless it has also received enough content from that user; more specifically, the absolute difference of uploading minus downloading from one user to another is bounded.

The introduction of such an incentive mechanism makes scheduling of information across a large distributed setting even more challenging. Given that nodes make decisions based on local information, a node may end-up downloading blocks that are already popular across the system and cannot be traded easily with other users. This effect gets amplified when the network frequently reconfigures. With network coding almost every block is unique and thus has higher chances of being useful to other users and being traded easily.

IV. Experimental evaluation

In this section we study the performance of an end-system cooperative architecture that uses network coding and compare it with other existing approaches. In particular we study the performance of end-system cooperative architectures for a) different types of topologies, b) heterogeneous client populations, c) dynamic node arrivals, d) sudden node and server departures, and d) incentive mechanisms.

To evaluate the performance of each scheme, we calculate the time it takes for each user to download the file. We are both concerned with the average download time, as well as the maximum, and the standard deviation of the waiting times among all clients. Another performance metric, is the overall utilization of the network, or, in other words, how fast the network can push information to the users. We measure network throughput as the total number of blocks transfered in a unit of time. This metric is also related to how much load is taken away from the server. The higher the throughput in the cooperative network, the more efficiently nodes are contributing to the download, and the lower the load in the server.

To study the performance of potentially large number of users under various settings and scenarios, we have implemented a simulator of an end-cooperative system that uses different algorithms for content distribution. Our purpose was not to construct a perfectly realistic simulation, but to demonstrate the advantages of network coding in some specific scenarios, which, we believe, are of practical importance.

Our simulator is used to compare the performance of content propagation using network coding, not coding at all, and coding only at the server. The input to the simulator is a set of nodes with constraints in their upload and download capacities, an initial overlay topology that connects these nodes, the size of the file to be distributed to the nodes, and the capacity of the single server in the system. The capacities are measured as the number of blocks that can be downloaded/uploaded in a single round. The number of blocks of the file transfered between two users is always an integral number. We have experimented with finer granularity and observed very similar results.

Whenever a user joins the system it picks four nodes at random and makes them its neighbors (provided that they have not exceeded the maximum number of neighbors, which is set to six in most of our experiments). The simulator supports dynamic user populations with nodes joining and leaving the system, and topology reconfigurations. In fact, at the end of each round, if a node determines that the utilization of its download capacity in the most recent rounds drops below a certain threshold (10% in most of our experiments), then it tries to discover and connect to new neighbors. Similarly, if the user has exceeded its maximum number of neighbors, then it will drop some of the old neighbors at random.

The simulator is round based. At the beginning of each round, each peer contacts its neighbors to discover whether there are new blocks that can be downloaded. For unencoded content and for source coding we assume that the each node knows the blocks available at its neighbors; for network coding we use the techniques described in Section III-C. Then, the simulator decides which blocks will be exchanged so that the upload and download capacities are not violated and that exchanges take priority as explained in Section III-D. Nodes with free download capacity contact the server. However, the number of nodes that can be satisfied by the server is bounded by the server's capacity. The block transfers, either from a peer or from the server, take place at the same round and, then, the system moves to the next round.

During each simulation all the nodes in the system use the same encoding scheme, either network coding, source coding, or propagating of original blocks.

To simulate a tit-for-tat scenario, the simulator keeps track of the difference between uploaded blocks minus downloaded blocks from a user S (source) to a user D (destination). If the difference is larger than the pre-configured value (typically 2 when we study tit-for-tat), then the source S will not send any block to D even if there is spare upload capacity at S and spare download capacity at D.

Obviously, there are important parameters that we do not simulate such as network delays, locality properties in constructing the overlay, cross-traffic impact, or malicious users. However, we believe that the simulator is able to capture some of the most important properties of an end-cooperative architecture.

Next we present the experimental results based on the simulator described above.

A. Homogeneous topologies

We start by comparing the performance of network coding (NC) to source coding (FEC) and unencoded information using a local rarest policy (LR) in a well-connected network of 200 nodes[3], where all nodes have the same access capacity equal to one block per round (homogeneous capacities). The upload capacity of the server is also 1 block/round. In this simulation we give priority to mutual exchanges, as described in Section III-D, but do not use the tit-for-tat mechanism.

In Figure 3, we plot the finish times of each node and the progress per round for that configuration. We measure the finish times as the number of rounds required to complete the download. We observe that in this baseline scenario all schemes perform equally well. The performance with network coding was slightly better compared to the other schemes, but, still, in all schemes the average finish time was close to the minimum finish time possible, which is rounds since the original file is equal to 100 blocks.

In the following sections we deviate from that baseline scenario and observe that with small changes in the configuration network coding performs better.

B. Topologies with clusters

Network coding has been shown to perform well in topologies with bad cuts. In this section we examine such a topology with two clusters of 100 nodes each. There is good connectivity and ample bandwidth between the nodes in each cluster (equal to 8 blocks per round in both directions and for all nodes). However there is limited bandwidth between the nodes of the different clusters; in every round only 4 blocks can propagate from one cluster to the other. The capacity of the server is also 4 blocks per round and, moreover, the server departs at round 30 and, thus, the nodes need to use the capacity between the clusters in the optimal way to retrieve the blocks of the file as fast as possible.

In Figure 4 we plot the finish times for each node. The minimum possible finish time in this experiment is equal to 25 rounds. Observe that without coding the average finish time is roughly three times longer compared to using network coding. The reason is that without coding some packets are transmitted multiple times over the bad cut wasting precious capacity that could have been used to transmit new blocks. For similar reasons, network coding outperforms source coding in this example by almost a factor of two. Given that nodes stay in the system after the download is complete and the server puts 20% extra erasure codes in the system, with source coding the chances of transmitting the same block multiple times over the cut are reduced. Thus, source coding performs much better than no coding, but still worse than network coding.

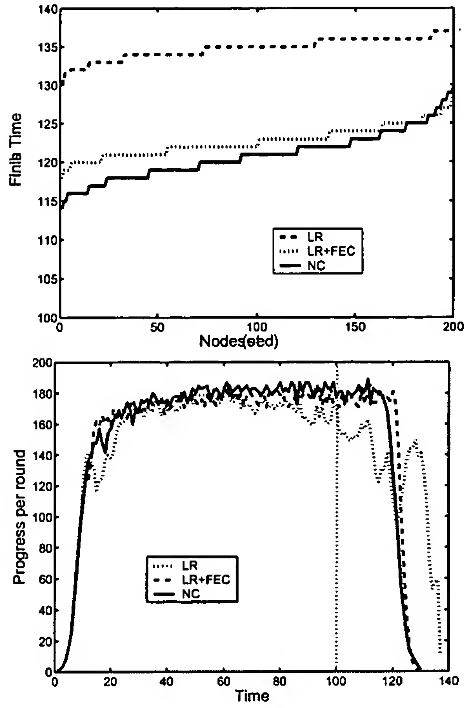

Fig. 3. Finish times and progress per round (measured in terms of the total number of blocks exchanged in the system) for a well-connected topology of 200 nodes. Size of the file is 100 blocks.

In this example, for both source coding and transmitting of original packets, we have assumed that nodes use the local rarest heuristic to decide which blocks to receive from their neighbors. Choosing random blocks gave worse finish times.

C. Heterogeneous capacities

We expect that the nodes of user collaborative distribution systems have non-homogeneous capabilities. The majority of the users are connected behind slow access links, including dial-up connections and ADSL connections, and a small percentage are connected with very fast links (e.g. corporate and university users). In this regard, we wish to study the performance experienced by fast users when they interact with other slower users in such heterogeneous environments.

Since fast users have more capacity, they are allowed to have more neighbors to fully utilize their extra capacity. However,

[3] We have performed limited experimentation with larger topologies of sizes up to 2000 nodes and observed similar results

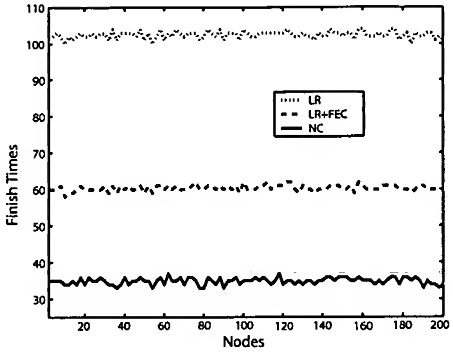

Fig. 4. Finish times of a topology with two clusters (100 nodes each).

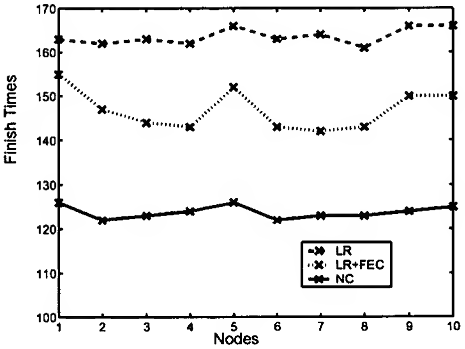

Fig. 5. Finish times of the fast nodes in a network with 10 fast nodes and 190 slow nodes. Size of the file is 400 blocks and capacity of server and fast nodes is 4 blocks/round.

the higher the number of neighbors feeding a fast node, the harder it is for the large set of neighbors to efficiently provide useful data in an uncoordinated fashion.

In Figure 5 we plot the finish times of the fast users in a network with many slow users and few fast users. In this example the fast users are 4 times faster than the slow users. We observe that with network coding the finish times of the fast nodes are on the average 20% better than source coding and around 30% better than with no coding. Also, observe that with network coding the difference in the observed finish time minus the minimum finish time is very similar to the baseline scenario of Figure 3, indicating that the heterogeneity did not affect the fast nodes with network coding, but had decreased the performance of the fast nodes with both source coding and no coding.

When network coding is not used, slow nodes may pick blocks from the server that are not needed by the fast nodes. Also slow nodes may use much of their capacity to share blocks that came from the fast nodes in the first place. The end result is that often the decisions taken by the many slow nodes do not take into account the interests of the fast nodes and fast nodes need to wait for many rounds to obtain the appropriate blocks.

TABLE I

FINISH TIMES FOR A FAST NODE AS THE RATIO OF THE CAPACITY OF THE FAST NODE OVER THE CAPACITY OF THE SLOW NODES INCREASES.

Method	x2	x4	x8
Random	107	166	281
Local Rarest	106	135	208
Source Coding Random	84	113	134
Source Coding LR	78	92	106
Global Rarest	75	92	98
Network Coding	69	72	73

Note: A ratio of x2 indicates that the capacity of the fast peer is two times the capacity of a slow peer. Similarly for x4 and x8. The number of slow peers is 50, 100, and 200 in the three cases respectively.

On the other hand, with network coding, the blocks that propagate in the network are linear combinations of many other blocks. Thus, the fast nodes have better chances of making progress in each round.

We have also noticed that as the capacity difference between fast nodes and slow nodes increases, fast nodes experience even worse performance when network coding is not used; on the other hand, with network coding, the performance degradation is minimal. In Table I we show some results that validate it.

In this experiment we have only one fast peer and many slow peers (50, 100, and 200 for the three cases), which allows us to focus our discussion; similar results exist for the case of a small subset of fast nodes. As the ratio of the capacity of the fast peer increases from 2 to 4 and to 8, we also increase the number of neighbors of the fast node (to give it the opportunity to use the extra capacity). We also scale accordingly the capacity of the server and the size of the file so that the minimum finish time for the fast node is 50 rounds. If no network coding is used, Table I shows a drastic increase in the finish times of the fast node as the capacities ratio increases. With network coding the finish time remains relatively unchanged indicating that heterogeneity in the capacities does not penalize the fast nodes. As a final note, the performance of the slow nodes across these experiments remain almost unchanged.

D. Dynamic arrivals and departures

Dynamic Arrivals

In this section we show the impact of dynamic arrivals in the performance of the system. When nodes arrive at different times, newly arriving nodes have different download objectives than the nodes that have been in the system for sometime. For instance, newly arriving nodes can benefit from any block while older nodes require a specific set of blocks to complement the blocks that they already have. However, since newly arriving nodes do not know about the exact needs of the other nodes in the system, they will often make download decisions that are of little use to existing nodes in the system. This gets reflected in Figure 6.

In Figure 6 we simulated a scenario where 40 empty nodes arrive every 20 rounds. The file size is 100 blocks. We assume that nodes stay in the system 10 more rounds after they finish

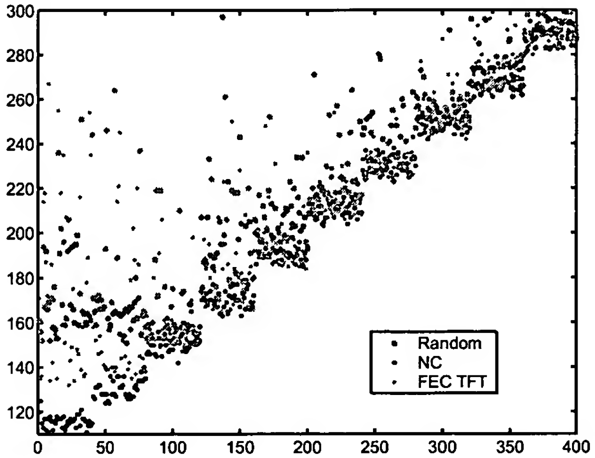

Fig. 6. Finish times for Random, Network Coding, and Source Coding (FEC) under dynamic arrivals. Nodes arrive in batches of 40 nodes every 20 rounds. Nodes stay in the system 10% extra rounds. Server stays forever. File size is 100 blocks.

the download and the server is always available. As we can see from Figure 6, the first set of nodes that arrived at time zero finish around time 110. In the ideal scenario where existing nodes are not delayed by the arrival of new nodes, 40 nodes should finish every 20 rounds. This is clearly the case with Network Coding.

However, when no encoding is used or when source coding is used, newly arriving nodes unnecessarily delay existing nodes. Existing nodes need to wait many extra rounds to receive useful information since newly arriving nodes spend much of their bandwidth performing download decisions that carry no information for existing nodes. Such difference is amplified for the first set of arriving nodes (e.g. a flash crowd). In this situation, network coding provides an improvement of 40% and 200% compared to source coding and no coding respectively. However, as time progresses the number of nodes that finish the download and stay around to help other nodes increases. As a result, the performance difference between network coding and the other approaches decreases roughly to 30%. Note, however, that this difference would increase substantially if nodes leave the system right after they finish their download.

Robustness to node departures.

When nodes are allowed to leave the system at any time and the server also leaves the system, then it is possible that some blocks disappear and reconstructing the original file is not possible (this is frequently observed in current file sharing P2P networks). This problem becomes noticeable when the blocks do not propagate evenly and, as a result, some blocks exist in only few nodes (rare blocks), and gets even more pronounced in dynamic networks, with frequent node departures. As we will see next, the inherent redundancy of network coding can help cope with such problems even in the most extreme cases of node departures. We quantify robustness in terms how finish times are impacted and in terms of how many nodes are able to complete the download.

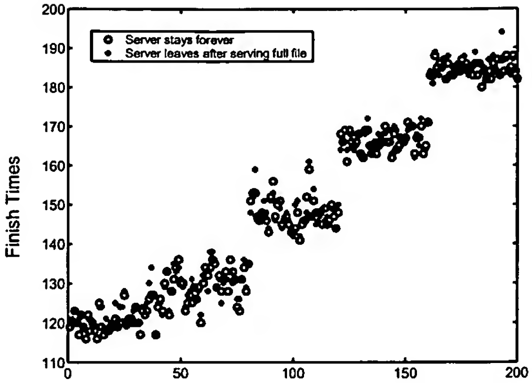

Fig. 7. Finish times for 200 nodes using network coding when a) the server stays for ever and b) when the server leaves after serving the full file. Nodes arrive in batches of 40 nodes every 20 rounds. Nodes leave immediately after downloading the file.

In Figure 7 we present the finish times of nodes using network coding when the server stays forever and compare it with the extreme scenario where the server leaves immediately after distributing one full copy of the file. In both cases the nodes depart from the system immediately after finishing the download. From Figure 7 we observe that with network coding the nodes finish roughly at the same time independently of the behavior of the server. Similarly, we have observed that the performance of network coding is independent of the amount of time that nodes stay around in the system after completing their download to other nodes with their downloads.

Another important feature of network coding is that even in extreme scenarios, all nodes are able to finish their downloads. For instance, in Figure 8, we show that if the server leaves the system after serving only 5% extra blocks, all 500 nodes in the system that use network coding complete the download. However, only 40% of the nodes finished downloading when source coding is used and only 10% of the nodes finished downloading when no coding was used.

In our experiments, we have observed that there is a certain server threshold after which most nodes are able to finish downloading in all scenarios. This threshold is around 10-15% extra rounds for source coding and 20-30% extra rounds for no coding. Notice, however, that the finish times will be significantly affected for the cases of source coding and no coding, whereas with network coding the performance perceived by the clients will not change significantly.

In summary, with network coding all nodes are able to finish downloading even in extremal scenarios; on the other hand, only a small percentage of nodes finishes downloading when source coding or no coding is used. Moreover, even in such extremal scenarios, the performance perceived by the users will not be affected significantly with network coding. Thus, network coding adds robustness to the system and it

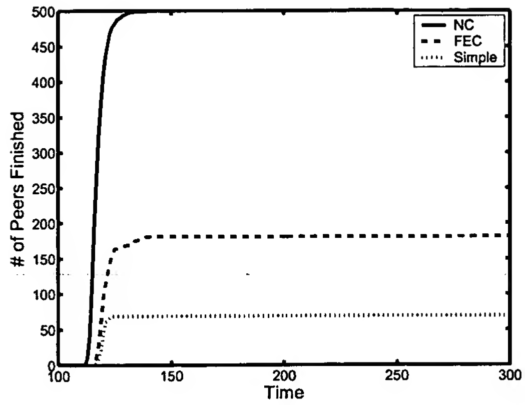

Fig. 8. Number of nodes finishing over time for random, source coding, and network coding. The network has 500 nodes. The server stays for 5% extra rounds. Nodes leave immediately after downloading the full file. Note that many nodes with random and source coding are not able to finish.

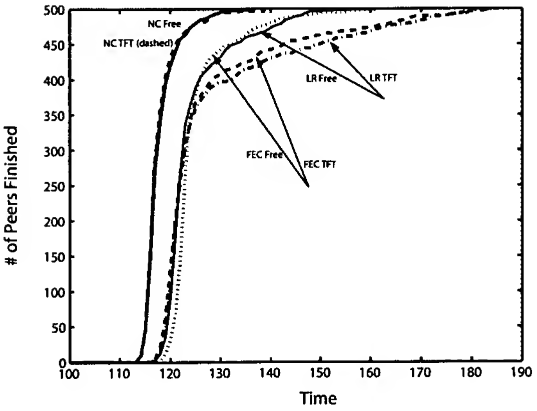

Fig. 9. Number of users finished by a given time (measured in number of rounds) for Network Coding (NC), Source Coding (FEC) and Local Rarest (LR) under a) no incentive schemes (Free) and b) tit-for-tat (TFT). Network size is 500 users. File size is 100 blocks.

is interesting to observe that the extra robustness is mostly introduced by the peers. This is opposed to source coding systems, which are dependent on the server to introduce extra encoded blocks and, hence, robustness.

E. Incentive mechanisms: Tit-for-tat

We have argued in Section III-D that the use of an incentive mechanism like tit-for-tat (TFT) may reduce the throughput of the system since blocks need to be carefully picked to be easily traded.

In Figure 9 we show the total number of peers finished by time *t* with and without the use of tit-for-tat to discourage free-riders. In this simulation the maximum allowable difference between blocks uploaded to a node minus the number of downloaded blocks from the same node is 2. In the case of network coding, the introduction of TFT has practically no observable impact on the performance of the system. However, for both source coding and no coding, the introduction of TFT significantly affects the finish times by delaying the upper tail of the distribution.

For instance, when transmitting unencoded blocks, the last user finished at time 161 without TFT and at time 185 with TFT. Similarly, when source coding was used the finish times were 159 and 182 respectively. The decrease in performance happens because nodes may end-up with blocks that are of little interest to their neighbors. Recall, however, that nodes are allowed to change neighbors if they are not able to receive enough throughput.

We have also experimented with larger networks and observed that increasing the size of the network amplifies the penalty introduced by using tit-for-tat, specially for a system where no-coding is used.

V. Summary and Further Directions

We propose a new content distribution system that uses network coding. Unlike other systems based on network coding, our approach targets the distribution of large files in a dynamic environment where nodes cooperate. Our system does not require any centralized knowledge of the network topology and nodes make decisions of how to propagate blocks of information based only on local information.

The main advantage of using network coding for distributing large files is that the scheduling of the content propagation in the overlay network is much easier. Deciding on the correct block of information to transmit to another node is difficult without global information; the transmitted packet is useful to the receiving node, but, may not be useful to other downstream nodes. With network coding, each generated block is a combination of all the blocks available to the transmitter and thus, if any of them is useful downstream, then the generated block will also be useful.

We have demonstrated through extensive simulations the performance advantages of using network coding over transmitting unencoded information and over coding at the source only in scenarios of practical interest. Network coding performs better when the nodes have heterogeneous access capacities, when the arrivals and departures of the nodes are not synchronized, when there are natural bottlenecks in the overlay topology (that need to be utilized as best as possible), and when incentive mechanisms are in place to discourage free-riders. The performance benefits provided by network coding in terms of throughput are more than 20-30% compared to coding at the server, and can be more than 2-3 times better compared to transmitting unencoded blocks. Moreover, we have observed that with network coding the system is much more robust to server and node departures.

Despite the rich literature in network coding, we are not aware of any operational content distribution network that uses network coding. Based on the system presented in this paper, we have implemented *Avalanche*, a real system using network coding. Through Avalanche, we are currently investigating the benefits of using network coding to distribute very large files to a large number of users in realistic settings.

During the design and implementation of Avalanche we have identified various practical issues related to network coding. The most important of them is the speed of encoding and decoding. Recall that during encoding we combine all available blocks and this is an $O(k)$ operation, where k is the number of the blocks of the file. During decoding, we invert a $k \times k$ matrix in $O(k^3)$ and then reconstruct the original file in $O(k^2)$. (In our system the reconstruction cost dominates the running time because it involves many reads of large blocks from the hard disk; whereas the inversion of the coefficient matrix can be done in memory.) We are currently considering techniques inspired from sparse codes to increase the encoding and decoding rates, which allow us to encode/decode files of more than 4 GB with very low overhead.

Another major concern in any content distribution scheme is the protection against malicious nodes. A malicious node can introduce arbitrary blocks in the system and make the reconstruction of the original file impossible. When the nodes do not perform coding, the server can digitally sign the packets transmitted and, thus, protect against malicious users. Digitally signing is more difficult when rateless codes are used, but recently [29] demonstrated how homomorphic collision-resistant hash functions can be used to provide protection in that case. Similar schemes can be used to provide protection when network coding is in place [30]. In Avalanche we use special sets of secure hash functions that survive network coding operations and consume very little computational resources, as opposed to traditional homomorphic hashes.

ACKNOWLEDGMENTS

We would like to thank our collaborators Philip A. Chou and Kamal Jain for their invaluable insights about Network Coding which greatly inspired the design of our system.

REFERENCES

[1] akamai, "Akamai," http://www.akamai.com.

[2] B. Cohen, "Incentives build robustness in bittorrent," in *P2P Economics Workshop*, Berkeley, CA, 2003.

[3] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, 2000.

[4] Philip Chou, Yunnan Wu, and Kamal Jain, "Network coding for the internet," in *IEEE Communication Theory Workshop*, Capri, 2004, IEEE.

[5] Paul Francis, "Yoid: Extending the internet multicast architecture," http://www.icir.org/yoid/docs/, 2000.

[6] Yang-hua Chu, Sanjay G. Rao, and Hui Zhang, "A case for end-system multicast," in *SIGMETRICS*, Santa Clara, CA, 2000, pp. 1–12, ACM.

[7] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowston, and Atul Singh, "Splitstream: High-bandwidth multicast in cooperative environments," in *19th Symposium on Operating Systems Principles (SOSP)*. 2003, ACM.

[8] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperating networking," in *NOSSDAV*, 2002.

[9] Vivek K. Goyal, "Multiple descriptive coding: Compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 73–93, 2001.

[10] Pablo Rodriguez-Rodriguez and Ernst Biersack, "Dynamic parallel-access to replicated content in the internet," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 455–465, 2002.

[11] John Byers, Michael Luby, and Michael Mitzenmacher, "Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads," in *Infocom*, 1999.

[12] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. Al Hamra, and L. Garces-Erice, "Dissecting bittorrent: Five months in a torrent's lifetime," in *Passive and Active Measurements (PAM)*, 2004.

[13] Dongyu Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *SIGCOMM*. 2004, ACM.

[14] Rob Sherwood, Ryan Braud, and Bobby Bhattacharjee, "Slurpie: A cooperative bulk data transfer protocol," in *Infocom*, Hong Kong, HK, 2004, IEEE.

[15] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *Symposium on Operating Systems Principles (SOSP)*, 2003.

[16] John Byers, Jeffrey Considine, Michael Mitzenmacher, and Stanislav Rost, "Informed content delivery across adaptive overlay networks," in *SIGCOMM*, Pittsburgh, PA, 2002, ACM.

[17] John Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege, "A digital fountain approach to reliable distribution of bulk data," in *SIGCOMM*, 1998.

[18] Petar Maymounkov and David Mazires, "Rateless codes and big downloads," in *IPTPS'03*, 2003.

[19] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing steiner trees," in *10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003.

[20] G. Robins and A. Zelikovsky, "Improved steiner tree approximation in graphs," in *7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2000.

[21] M. Thimm, "On the approximability of the steiner tree problem," *Mathematical Foundations of Computer Science*, vol. 295, no. 103, pp. 387–402, 2001.

[22] Ying Zhu, Baochun Li, and Jiang Guo, "Multicast with network coding in application-layer overlay networks," *IEEE Journal on Selected Areas in Communications*, 2004.

[23] P. A. Chou, Y. Wu, and K Jain, "Practical network coding," in *Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2003.

[24] Kamal Jain, Laszlo Lovasz, and Philip Chou, "Building scalable and robust peer-to-peer overlay networks for broadcasting using network coding," Tech. Rep., Microsoft Research, 2004, Available at ftp://ftp.research.microsoft.com/pub/tr/TR-2004-135.pdf.

[25] G. Pandurangan, P. Raghavan, and E. Upfal, "Building low-diameter p2p networks," in *42nd Annual Symposium on Foundations of Computer Science (FOCS01)*, 2001, pp. 492–499.

[26] Reza Rejaie and Shad Stafford, "A framework for architecting peer-to-peer receiver-driven overlays," in *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSDAV)*, Cork, Ireland, 2004, pp. 42–47, ACM Press.

[27] Tracey Ho, Ralf Koetter, Muriel Medard, David R. Karger, and Michelle Effros, "The benefits of coding over routing in a randomized setting," in *IEEE International Symposium on Information Theory (ISIT)*, Yokohama, Japan, 2003, p. 442, IEEE.

[28] Eytan Adar and Bernando A. Huberman, "Free riding on gnutella," *First Monday*, 2000.

[29] Maxwell N. Krohn, Michael J. Freedman, and David Mazires, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *IEEE Symposium on Security and Privacy*, Berkeley, California, 2004, pp. 226–241, IEEE.

[30] Christos Gkantsidis and Pablo Rodriguez Rodriguez, "Cooperative security for network coding file distribution," Tech. Rep. MSR-TR-2004-137, Microsoft Research, 2004.